Лабораторная работа №4
Базовые растровые алгоритмы

Постановка задачи

Целью работы являлась разработка приложения для визуализации и анализа базовых растровых алгоритмов. Требовалось реализовать следующие алгоритмы:

- Пошаговый алгоритм растеризации отрезков
- Алгоритм ЦДА (Digital Differential Analyzer)
- Алгоритм Брезенхема для отрезков
- Алгоритм Брезенхема для окружностей
- Алгоритм Ву для сглаженных линий
- Алгоритм Кастла-Питвея

Теоретическая часть

Основы растровой графики

Растеризация - процесс преобразования векторных данных в растровое изображение. Основная задача - аппроксимация идеальных геометрических примитивов дискретными пикселями.

Математические основы алгоритмов

Пошаговый алгоритм использует уравнение прямой:

$$y = kx + b, \quad k = \frac{y_2 - y_1}{x_2 - x_1}$$

Для каждой координаты x вычисляется соответствующая координата y.

Алгоритм ЦДА основан на дифференциальном уравнении:

$$\frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1} = k$$

Приращения координат:

$$\Delta x = \frac{dx}{steps}, \quad \Delta y = \frac{dy}{steps}, \quad steps = \max(|dx|, |dy|)$$

Алгоритм Брезенхема использует целочисленную арифметику через анализ ошибки:

$$err = 2\Delta y - \Delta x$$

На каждом шаге:

$$err = \begin{cases} err + 2\Delta y, & \text{если } err < 0 \\ err + 2(\Delta y - \Delta x), & \text{иначе} \end{cases}$$

Алгоритм By реализует сглаживание через взвешенное окрашивание пикселей:

$$I_{pixel} = \alpha \cdot I_{line} + (1 - \alpha) \cdot I_{background}$$

где α - коэффициент, пропорциональный расстоянию до идеальной линии.

Алгоритм Кастла-Питвея основан на алгоритме Евклида и строит последовательность шагов через рекуррентные соотношения строк.

Реализация алгоритмов

Приложение разработано на C# с использованием Windows Forms. Основные компоненты:

- **Form1** - основная форма с элементами управления
- **Panel** - область для рисования с системой координат
- **ComboBox** - выбор алгоритма растеризации
- **NumericUpDown** - управление масштабом и толщиной линии

Система координат

Реализована декартова система координат с центром в середине панели:

$$screenX = centerX + logicalX \times scale$$

$$screenY = centerY - logicalY \times scale$$

Основные методы

```
- StepByStepAlgorithm() - пошаговый алгоритм
- DDAAlgorithm() - алгоритм ЦДА
- BresenhamLineAlgorithm() - алгоритм Брезенхема
- BresenhamCircleAlgorithm() - алгоритм Брезенхема для окружностей
- WuLineAlgorithm() - алгоритм By со сглаживанием
- CastlePitwayAlgorithm() - алгоритм Кастла-Питвея
- DrawPixel() - отрисовка пикселя с учетом масштаба
```

Сравнительная характеристика алгоритмов

Детальный анализ характеристик

Временные характеристики

Измерение времени выполнения проводилось с использованием класса Stopwatch. Результаты показывают значительное преимущество алгоритма Брезенхема благодаря использованию исключительно целочисленной арифметики.

Таблица 1: Сравнительная таблица алгоритмов растеризации

Сравнительная характеристика алгоритмов				
Название алгоритма	Тип операций	Скорость	Качество	Применение
Пошаговый алгоритм	Вещественные	Низкая	Среднее	Учебные цели
Алгоритм ЦДА	Вещественные	Средняя	Среднее	Простая графика
Алгоритм Брезенхема (отрезок)	Целочисленные	Высокая	Хорошее	Основная графика
Алгоритм Брезенхема (окружность)	Целочисленные	Высокая	Хорошее	Кривые 2 порядка
Алгоритм Ву	Вещественные	Низкая	Отличное	Качественная визуализация
Алгоритм Кастла-Питвея	Целочисленные	Средняя	Хорошее	Академические задачи

Сравнение производительности

- **Брезенхем** - самый быстрый благодаря целочисленным операциям
- **ЦДА** - средняя скорость из-за вещественных операций
- **Пошаговый** - низкая скорость из-за многократного округления
- **Ву** - низкая скорость из-за сложных вычислений прозрачности
- **Кастла-Питвея** - низкая скорость из-за работы со строками

Примеры работы алгоритмов

Качество визуализации

Алгоритм Брезенхема демонстрирует оптимальное сочетание скорости и качества для большинства практических задач.

Алгоритм Ву обеспечивает наилучшее качество за счет сглаживания, но требует значительных вычислительных ресурсов.

Алгоритм Кастла-Питвея представляет академический интерес благодаря математической элегантности, но не эффективен для практического применения.

Трудности при выполнении работы

1. **Преобразование координат** - обеспечение точного соответствия между логическими координатами и позициями пикселей на экране
2. **Реализация алгоритма Ву** - корректное вычисление коэффициентов прозрачности и смешивание цветов

Таблица 2: Технические характеристики алгоритмов

Технические характеристики алгоритмов				
Название алгоритма	Сложность	Точность	Особенности	Ограничения
Пошаговый алгоритм	O(n)	Ошибки округления	Простота реализации	Только отрезки
Алгоритм ЦДА	O(n)	Равномерное распределение	Улучшенный пошаговый	Вещественная арифметика
Алгоритм Брезенхема (линия)	O(n)	Абсолютная точность	Оптимальное быстродействие	Только отрезки
Алгоритм Брезенхема (окружность)	O(R)	Симметричность	8-точечная симметрия	Только окружности
Алгоритм Ву	O(n)	Антиалиасинг	Взвешенное окрашивание	Высокие затраты
Алгоритм Кастла-Питвея	O(n log n)	Математическая точность	Рекуррентная структура	Низкая эффективность

3. **Алгоритм Кастла-Питвея** - правильная интерпретация рекуррентных соотношений и построение последовательности шагов

4. **Оптимизация перерисовки** - эффективное обновление canvas при изменении масштаба и параметров отображения

Результаты

Разработанное приложение успешно реализует все требуемые алгоритмы растеризации. Основные достижения:

- Корректная работа всех шести алгоритмов растеризации
- Удобный интерфейс для сравнения различных методов
- Система координат с масштабированием и сеткой
- Измерение временных характеристик алгоритмов
- Визуализация результатов с различными цветами для каждого алгоритма

Приложение может использоваться для учебных целей и как инструмент анализа эффективности растровых алгоритмов.